OBSERVING AND STEERING AFFORDANCE IN VLA HIDDEN STATES FOR SPATIAL ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) policies achieve strong performance on manipulation benchmarks, but their success rate degrades sharply once object positions drift outside the training distribution, limiting real-world deployment. We ask whether a VLA's internal representations already encode a high-level semantic feature that reflects task progress, and whether this feature can be observed and controlled at inference time, without retraining the policy. We show that a linear probe trained on VLA hidden states reliably decodes an affordance cost derived from a 3D affordance cost field, and that this cost can be driven toward a target value at inference time through probe-derived activation perturbations. A layer-wise causal analysis identifies where perturbations both lower the predicted cost and change the resulting action, guiding the intervention layers evaluated in the OOD benchmark. We evaluate this observe-and-control approach on two recent open-source VLA backbones, OpenVLA and $\pi_{0.5}$, under a simulated closed-loop out-of-distribution benchmark built on LIBERO-Pro, in which the target object is displaced by an increasing radius from its training position. Without modifying any policy weights, steering raises out-of-distribution task success from 16.6% to 29.5% on OpenVLA and from 34.5% to 41.1% on $\pi_{0.5}$, an average absolute improvement of nearly 10 percentage points across both backbones.

1 INTRODUCTION

Vision-Language-Action (VLA) models map visual inputs and natural language instructions directly to low-level robotic controls (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Despite strong in-distribution performance, these policies exhibit fragile robustness once deployed out-of-distribution (OOD) (Ross et al., 2011; de Haan et al., 2019)—most notably when the target object is spatially displaced from its training layout. We hypothesize that this bottleneck stems from the VLA's inability to form a robust internal representation of the target object's *affordance*—the actionable properties dictating where and how it can be interacted with—so that, instead of an invariant grounding of functional object regions, the policy relies on spurious spatial shortcuts that collapse under displacement, yielding erratic, goal-divergent actions (Xu et al., 2026; Pang et al., 2025; Li, 2025).

We diagnose and rectify this vulnerability by treating the VLA's intermediate hidden states as the causal mediator between perception and action, and ask two coupled questions: can this affordance be linearly **decoded** from hidden states, and, if so, can it also be **controlled**—perturbed at inference time to steer the policy's physical actions? For decoding, we construct a continuous 3D *affordance cost field* over the workspace (Huang et al., 2023) and train a linear probe on the VLA's hidden states to predict the affordance cost at the robot's *current end-effector position*; the probe fits this field with exceptional fidelity, showing that VLA hidden states implicitly track the end-effector–target affordance relationship. For control, we derive an *activation-steering* direction from the probe coefficients in hidden-state coordinates (Turner et al., 2023) and inject it into the hidden states whenever the predicted cost signals an unpromising trajectory, causally redirecting the end-effector toward the target. This training-free intervention repairs erroneous behavior in previously unviable OOD scenarios; we validate it across two architecturally distinct backbones, OpenVLA (Kim et al., 2024) and $\pi_{0.5}$ (Physical Intelligence et al., 2025), with substantial success-rate gains and no fine-tuning.

Our main contributions are summarized as follows:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Decodable Affordance Representations**: Spatial object affordances are linearly decodable from VLA hidden states. On OpenVLA, our probe reaches R^2=0.893 across 84,653 timesteps, a property that also holds on $\pi_{0.5}$.
- **Causal Activation Steering**: We present *probe-direction bisect steering*, an inference-time intervention with adaptive trigger gating (threshold, slope, cooldown) and binary-search strength selection along a probe-derived cost-decreasing direction.
- **Extensive OOD Benchmarking**: Across 10 LIBERO-Object tasks under a circle-displacement protocol, diagnostic-guided steering raises aggregate success from 16.6% to 29.5% on OpenVLA and from 34.5% to 41.1% on $\pi_{0.5}$.

2 Related Work

Vision-language-action manipulation. OpenVLA (Kim et al., 2024) predicts discrete action tokens on LIBERO (Liu et al., 2023), whereas $\pi_{0.5}$ (Physical Intelligence et al., 2025) uses a continuous flow-matching action head; OpenVLA-OFT (Kim et al., 2025) is another continuous-action variant. We test our intervention on the architecturally distinct OpenVLA and $\pi_{0.5}$ backbones.

OOD robustness benchmarks. LIBERO-PRO (Zhou et al., 2025) and SimplerEnv (Li et al., 2025) assess controlled layout and simulation-to-real shifts, respectively. LIBERO-Plus (Fei et al., 2025) documents VLA failures under perturbations, while INT-ACT (Fang et al., 2025) identifies an intention–execution gap. Our circle-displacement protocol evaluates spatial robustness on LIBERO-Object.

Affordance representations. VoxPoser (Huang et al., 2023) constructs 3D value maps for manipulation; ReKep (Huang et al., 2024) optimizes relational keypoint constraints. AFI (Xu et al., 2026) reranks waypoints using an externally reconstructed affordance field, while RoboPoint (Yuan et al., 2024) predicts spatial affordance keypoints. We use affordance cost fields to train hidden-state probes on in-distribution rollouts, then steer activations in displaced-object scenes.

Probing and activation steering. Probes reveal linearly decodable information (Belinkov, 2022); representation engineering and contrastive activation addition steer behavior along learned directions (Zou et al., 2023; Panickssery et al., 2024). Recent VLA studies also intervene on activations (Turner et al., 2023; Buurmeijer et al., 2026; Swann et al., 2026). Our cost probe serves as both readout and steering direction, with online magnitude selection.

Probe-guided correction for VLAs. Closest concurrent methods decode target/grasp coordinates and apply post-hoc correction (Park et al., 2026; Zhang et al., 2026a), detect failure (Gu et al., 2025), or rerank sampled actions using a probed value signal (Zhang et al., 2026b). Instead, we steer the hidden states inside the forward pass, before an action is emitted.

3 Method

In this section, we detail our observe-and-steer framework for VLA models. Our approach operates in two phases: an **observation** phase, in which we construct a 3D affordance cost field and train linear probes to decode its cost at the end-effector position directly from the VLA's hidden states; and a **control** phase, in which we leverage these probes to causally steer the internal representations toward lower-cost states during deployment, correcting the robot's trajectory on the fly.

3.1 Preliminary

A VLA policy π_{VLA}, built on a pretrained transformer backbone (Vaswani et al., 2017), maps a task instruction u and visual observation I_t to a robot action a_t: $a_t = \pi_{\mathrm{VLA}}(I_t, u)$. Here, t indexes policy inference steps, I_t is the current visual observation, and u remains fixed throughout the episode. The frozen backbone has $L+1$ decoder layers indexed by $\ell \in \{0, \ldots, L\}$, where L denotes the terminal-layer index. At each layer, $\mathbf{h}_{t,\ell} \in \mathbb{R}^D$ is the hidden vector at the last prompt-token position, with hidden dimension D.

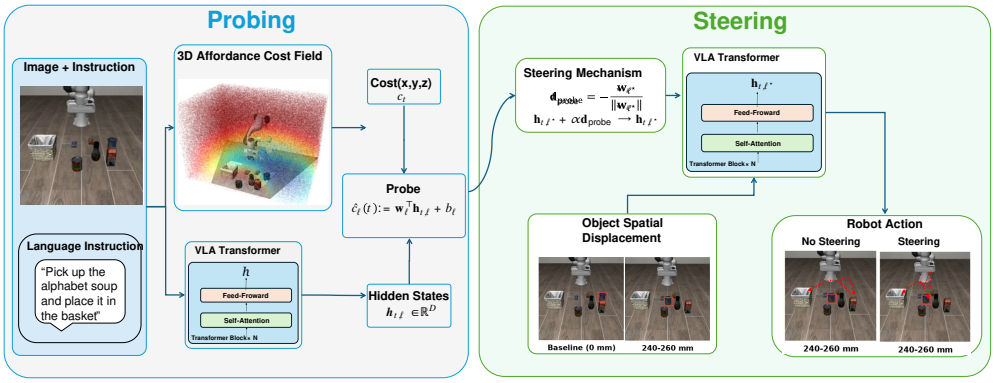


Figure 1: **Observe-and-steer overview.** *Probing* (left): a linear probe decodes the end-effector's affordance cost from the VLA's hidden states, supervised by a 3D affordance cost field built from the scene. *Steering* (right): under object displacement, the probe's negative weight direction is added back into the hidden state at a selected layer to redirect the end-effector toward the target. The base policy remains frozen throughout.

We hypothesize that, by jointly processing the task instruction u and the current visual observation I_t, the VLA understands both the task goal and the scene, and that its intermediate hidden states $\mathbf{h}_{t,\ell}$ perceive the *affordance* of the robot's current end-effector—i.e., how favorable its spatial position is for interacting with the target object and completing the task. Based on this, we pursue two consequences: that this affordance can be linearly **decoded** from $\mathbf{h}_{t,\ell}$, and that it can be actively **controlled** by perturbing $\mathbf{h}_{t,\ell}$ at inference time to steer a_t toward lower-cost, corrective behavior—the subjects of Observability and Control below, respectively.

3.2 Observability: Linear Affordance Probes

The goal of the probe is to predict the affordance cost of the robot's current end-effector position from the VLA's hidden states $\mathbf{h}_{t,\ell}$. To this end, we first describe how to construct a 3D affordance cost field for the current task scene that provides supervised labels for probe training, and then present the probe training procedure together with the resulting experimental findings.

Affordance Cost Field Construction

We assign each voxel in the 3D workspace a scalar affordance cost $c \in [0, 1]$, where a lower value indicates a more favorable end-effector position for the current subtask. Following (Huang et al., 2023; Xu et al., 2026), we construct this field in two stages: first grounding the semantic target into a 3D point cloud, then converting it into a continuous voxel-wise cost field.

Given the task instruction u and the current RGB observation, a powerful vision-language model decomposes u into temporally ordered sub-goals (e.g., pick $\rightarrow$ place) and extracts the target token for the active stage (e.g., the object to grasp, or the receptacle to place into). An open-vocabulary segmentation model (e.g., Grounded-SAM) then produces a 2D mask of this target; combining the mask with the depth map and camera intrinsics, we back-project the masked pixels into 3D space to obtain a target point cloud $\mathcal{P}_{\text{target}}$, while the full RGB-D observation yields the scene point cloud $\mathcal{P}_{\text{scene}}$. Because pick-and-place tasks change their semantic focus across stages, we build one field per stage at episode initialization (stage 1: approach the object while avoiding the receptacle; stage 2: the roles are reversed), and a lightweight online detector switches the active field when task progress advances.

We then discretize the robot workspace into a $K \times K \times K$ voxel grid, with K voxels per axis, and project both point clouds onto it. Within this grid we compute two complementary subfields: a *target guidance field* V_{target}, obtained by a Euclidean distance transform to the target centroid so that cost decreases near the goal, and an *obstacle avoidance field* V_{obst}, which assigns high cost to voxels occupied by or near $\mathcal{P}_{\text{scene}}$. The spatial affordance field (SAF), used here as an affordance

cost field, is their weighted linear combination,

$$V_{\mathrm{SAF}} = w_{\mathrm{target}}\, V_{\mathrm{target}} + w_{\mathrm{obst}}\, V_{\mathrm{obst}}, \tag{1}$$

with combination weights $w_{\mathrm{target}}{:}w_{\mathrm{obst}} = 2{:}1$. We then apply Gaussian smoothing and normalize the field to $[0, 1]$. The resulting detector-guided cost c_t is the lookup of V_{SAF} at the current end-effector position under the active stage, and serves as the supervised label for probe training and online monitoring.

Figure 2 visualizes the ground-truth affordance cost c_t of the end-effector along successful and failed trajectories over time, for this two-stage task. On successful rollouts, c_t decreases within each stage as the end-effector progresses toward the corresponding subgoal; on failed rollouts, by contrast, the cost does not converge monotonically and instead exhibits oscillations—confirming that c_t tracks genuine task progress and is a meaningful supervised target for the probe trained next.

PROBE TRAINING

For each backbone (OpenVLA and $\pi_{0.5}$), the probe dataset comprises 500 in-distribution trajectories (50 per task) from the 10-task LIBERO-Object suite (Liu et al., 2023). At each recorded timestep, we pair the layer-wise hidden states with the end-effector affordance cost. We split the data into training and validation sets by complete trajectory, so that no trajectory contributes timesteps to both sets. Each layer's training set contains N timestep samples indexed by j:

$$\mathcal{D}_\ell = \left\{ (\mathbf{h}_\ell^{(j)},\, c^{(j)}) \right\}_{j=1}^{N}. \tag{2}$$

For each layer ℓ, we standardize hidden features as $\mathbf{z}_{t,\ell} = (\mathbf{h}_{t,\ell} - \boldsymbol{\mu}_\ell) \oslash \boldsymbol{\sigma}_\ell$, where $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\sigma}_\ell$ are the per-feature training mean and standard deviation, and $\oslash$ denotes element-wise division. In the original hidden-state coordinates, the fitted probe takes the form shown in Figure 1:

$$\hat{c}_\ell(t) := \mathbf{w}_\ell^\top \mathbf{h}_{t,\ell} + b_\ell, \tag{3}$$

with coefficients obtained by fitting ridge regression on standardized samples,

$$(\widetilde{\mathbf{w}}_\ell, \widetilde{b}_\ell) = \arg\min_{\mathbf{v},\beta} \sum_{j=1}^{N} \left(c^{(j)} - \mathbf{v}^\top \mathbf{z}_\ell^{(j)} - \beta \right)^2 + \lambda \|\mathbf{v}\|_2^2, \tag{4}$$

where $\mathbf{v}$ and β are the regression coefficients and intercept optimized during fitting, and the regularization strength λ is selected by cross-validation on the training split. The coefficients in Eq. 3 absorb feature standardization: $\mathbf{w}_\ell = \widetilde{\mathbf{w}}_\ell \oslash \boldsymbol{\sigma}_\ell$ and $b_\ell = \widetilde{b}_\ell - \mathbf{w}_\ell^\top \boldsymbol{\mu}_\ell$. Validation scores are computed on held-out trajectories. The probes remain fixed during OOD evaluation with displaced target objects.

Probe Results Figure 3 reports the validation coefficient of determination (R^2) and mean absolute error (MAE) of the affordance probe at every decoder layer for both OpenVLA and $\pi_{0.5}$. The results confirm our hypothesis that affordance is linearly decodable from hidden states on both backbones, with comparable best-layer accuracy (R^2=0.893 at OpenVLA L4; R^2=0.916 at $\pi_{0.5}$ L0), though *where* it is most decodable differs—OpenVLA shows a broad shallow plateau, while $\pi_{0.5}$ peaks at the first layer and decays with depth.

3.3 CONTROL: PROBE-DIRECTION STEERING

At inference time, we use the fixed probe learned from in-distribution rollouts to steer hidden states produced by displaced-object observations toward lower predicted cost.

Steering mechanism. At each eligible inference step, we modify the hidden state at a chosen intervention layer $\ell^\star$ by adding a perturbation along the direction that decreases the probe's predicted affordance cost:

$$\mathbf{h}_{t,\ell^\star} \leftarrow \mathbf{h}_{t,\ell^\star} + \alpha\, \mathbf{d}_{\mathrm{probe}}, \tag{5}$$

where $\alpha > 0$ is the perturbation magnitude and $\mathbf{d}_{\mathrm{probe}} = -\mathbf{w}_{\ell^\star} / \|\mathbf{w}_{\ell^\star}\|_2$ is the normalized negative gradient of the intervention-layer probe. The perturbation is applied only during the prefill forward

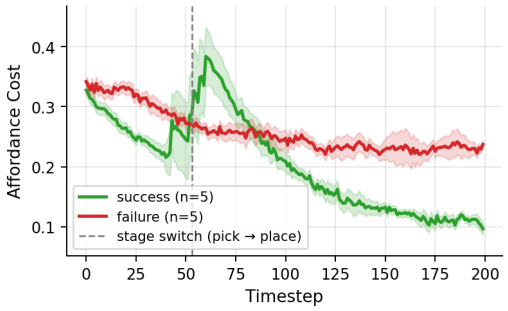


Figure 2: Mean end-effector affordance cost c_t over time for successful and failed OpenVLA rollouts from LIBERO. Successful trajectories decrease within each stage; failed ones oscillate. Gray dashed line: mean stage 1→2 transition.

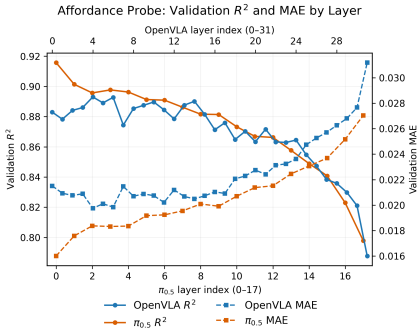


Figure 3: Validation R^2 (circles, left axis) and MAE (squares, right axis) of the affordance probe across decoder layers, for OpenVLA (blue, top axis) and $\pi_{0.5}$ (orange, bottom axis).

pass, at the last prompt token position. Because the transformer continues to propagate this modification through all subsequent layers, a perturbation at an intermediate layer $\ell^\star < L$ can alter the terminal-layer representation—and thus the final action output—without requiring a separate gradient computation, connecting to the broader probe-direction steering lineage discussed in Related Work (Turner et al., 2023; Li et al., 2023).

Intervention gating. Continuous steering at every step can destabilize the VLA's low-level motor control, so we intervene only when the predicted trajectory appears unpromising. Three conditions gate each intervention and are checked in sequence; the effect of these gating hyperparameters is analyzed in the Experiments section:

- **Threshold (τ).** Raw per-step predictions $\hat{c}_L(t)$ are noisy, so we track an exponential moving average (EMA) $\bar{c}_t$ with smoothing coefficient α_{ema} and consider intervention only when $\bar{c}_t$ exceeds a threshold τ. If the smoothed predicted cost is already low, the end-effector is likely progressing toward the subgoal and intervention is unnecessary.

$$\bar{c}_t = \alpha_{\text{ema}}\, \hat{c}_L(t) + (1 - \alpha_{\text{ema}})\, \bar{c}_{t-1}. \tag{6}$$

- **Slope (δ).** Even when $\bar{c}_t$ is high, the policy may already be recovering. We therefore compute the change $\hat{c}_L(t) - \hat{c}_L(t - W)$ over a window of W replans; if this quantity falls below δ, the cost is declining fast enough that the robot is self-correcting, and we skip the intervention.
- **Cooldown (κ).** Finally, we enforce a minimum interval of κ replans between consecutive interventions, preventing high-frequency perturbations from oscillating the action sequence and overwriting the VLA's short-horizon motor plans.

Intervention layer selection. The choice of $\ell^\star$ is non-trivial because an effective intervention must both lower the predicted affordance cost and change the resulting action: deep-layer perturbations tend to reduce the terminal probe cost while leaving action tokens unchanged, whereas shallow-layer perturbations alter actions without reliably reducing the probe readout. Figure 4 reports this dissociation on $\pi_{0.5}$. This is consistent with brief layer-wise analyses reported for other VLA steering methods: FFN-vector interventions are most effective at late or full-network depth rather than early layers (Häon et al., 2025), and linear interventions on $\pi_{0.5}$ and OpenVLA representations weaken with depth as representation norm grows (Buurmeijer et al., 2026). An offline layer–magnitude analysis jointly examines whether perturbations move the terminal-layer probe prediction toward a low-cost target c_{target} and induce non-trivial changes in the emitted action. This diagnostic prioritizes candidate layers for OOD evaluation. Details are given in the Appendix.

Strength selection. Having chosen the intervention layer $\ell^\star$, we still need to set the perturbation magnitude α. Our criterion is to drive the *last-layer* probe prediction toward a pre-specified low-cost

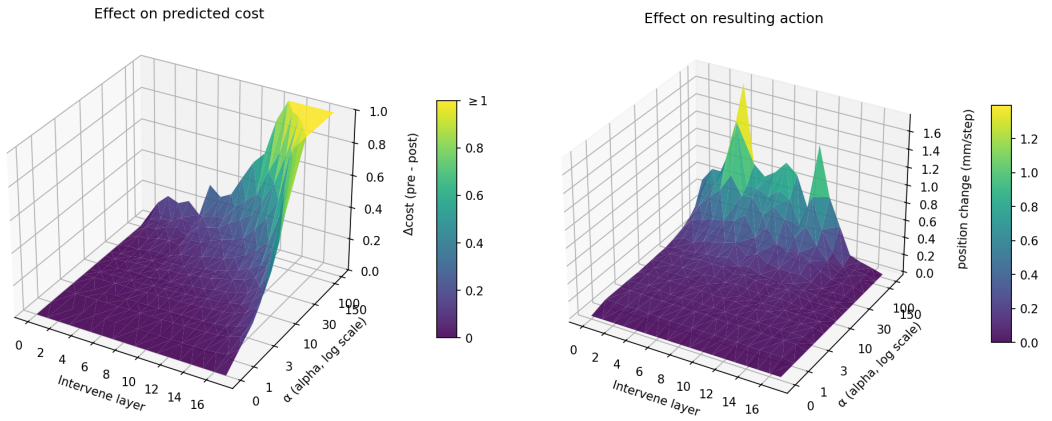


Figure 4: Intervention layer and perturbation magnitude analysis on $\pi_{0.5}$ (in-distribution, vanilla observations). Left: change in predicted cost; colorbar saturates at $\Delta\text{cost} = 1$. Right: resulting end-effector position change (mm/step).

Algorithm 1 Intervention Gating (per inference step)

Input: hidden state $\mathbf{h}$, probe at $\ell^\star$, terminal-layer probe at L

$\hat{c}_L(t) \leftarrow \textsc{Probe}_L(\mathbf{h})$
update $\bar{c}_t$ via Eq. 6
if $\bar{c}_t \leq \tau$ **then**
  return no intervention {threshold}
end if
if $\hat{c}_L(t) - \hat{c}_L(t - W) < \delta$ **then**
  return no intervention {slope}
end if
if fewer than κ replans since the last intervention **then**
  return no intervention {cooldown}
end if
$\alpha^\star \leftarrow \textsc{SelectStrength}(\hat{c}_L, c_{\text{target}})$
Inject $\alpha^\star \, \mathbf{d}_{\text{probe}}$ at layer $\ell^\star$ during prefill

target c_{target}, obtained offline by analyzing successful trajectories (more details in the Appendix): for a candidate α, we re-run the prefill forward pass with the perturbation injected at $\ell^\star$, read out $\hat{c}_L(\alpha)$ from the layer-L probe, and search

$$\alpha^\star \approx \arg\min_{\alpha} \left| \hat{c}_L(\alpha) - c_{\text{target}} \right| \tag{7}$$

via bisection. If increasing α raises $\hat{c}_L$ rather than lowering it, the direction is insensitive at that step and the search is skipped. Figure 5 plots this dose–response: $\hat{c}_L(\alpha)$ falls with α most steeply at layers near the readout—so a small α already reaches c_{target}—whereas shallower layers require a much larger α for the same effect.

Algorithm 1 summarizes the resulting per-step online control procedure: the three gating conditions above are checked in order, and any one of them being met skips the intervention for that step; otherwise the perturbation strength $\alpha^\star$ is set by the search described above.

4 Experiments

In this section, we first describe the experimental setup, then report closed-loop task success under out-of-distribution object displacement, and finally present hyperparameter analyses of the intervention gating.

Table 1: OpenVLA OOD success (%): steering / baseline, 20 trials per task–bin cell.

	Target displacement (mm)								
Task	0–20	20–40	40–60	60–70	70–80	80–85	85–90	90–95	Total
Soup	100/60	75/35	35/10	20/0	15/0	15/10	5/0	5/5	34/15
Cheese	75/75	80/55	45/20	25/0	15/5	10/0	5/0	5/0	32/19
Salad	100/85	65/55	30/0	10/0	5/0	10/0	0/0	5/0	28/18
BBQ	85/65	85/35	45/30	25/5	10/5	5/5	10/5	5/0	34/19
Ketchup	80/45	35/10	10/0	5/0	5/0	0/0	0/0	0/0	17/7
Tomato	80/70	50/35	20/5	5/0	5/0	0/0	5/0	0/0	21/14
Butter	100/90	90/55	65/10	20/0	15/0	5/0	0/0	10/0	38/19
Milk	100/90	60/40	35/5	15/0	10/5	0/0	0/0	5/0	28/18
Choc.	100/90	55/45	15/15	25/0	20/0	40/5	25/5	10/5	36/21
Juice	100/90	50/30	30/5	15/5	5/0	10/5	5/0	0/0	27/17
Aggregate	**92.0/76.0**	**64.5/39.5**	**33.0/10.0**	**16.5/1.0**	**10.5/1.5**	**9.5/2.5**	**5.5/1.0**	**4.5/1.0**	**29.5/16.6**

Table 2: $\pi_{0.5}$ OOD success (%): steering / baseline, 20 trials per task–bin cell.

	Target displacement (mm)							
Task	140–160	160–180	180–200	200–220	220–240	240–260	260–280	Total
Soup	90/90	65/65	65/60	60/60	70/55	55/45	40/30	64/58
Cheese	90/80	80/60	75/55	80/60	70/55	40/35	40/40	68/55
Salad	45/40	35/35	35/25	20/15	15/15	20/15	0/0	24/21
BBQ	85/85	70/65	55/45	45/45	35/35	10/10	0/0	43/41
Ketchup	40/45	40/30	20/25	15/5	10/0	5/0	0/0	19/15
Tomato	70/65	75/60	65/55	45/45	45/35	20/5	5/5	46/39
Butter	40/25	40/25	45/30	55/45	50/45	25/15	20/15	39/29
Milk	70/65	75/60	20/15	10/5	5/0	5/0	0/0	26/21
Choc.	45/45	35/30	45/40	30/20	25/20	20/5	5/5	29/24
Juice	100/95	85/75	65/50	50/40	40/30	20/15	10/5	53/44
Aggregate	**68/64**	**60/50**	**49/40**	**41/34**	**36/29**	**22/14**	**12/10**	**41.1/34.5**

4.1 Datasets and Settings

We evaluate closed-loop steering on LIBERO-Object using the fixed probes trained in Section 3.2. OOD evaluation trajectories do not contribute to probe fitting. For steering, we build an OOD evaluation set whose primary shift is a displacement of the target object from its nominal training position, following the position-perturbation protocol of LIBERO-PRO (Zhou et al., 2025) (Figure 6): displacement is discretized into bins—eight spanning 0–95 mm for OpenVLA, seven spanning 140–280 mm for $\pi_{0.5}$—with 20 randomly sampled target positions per bin, yielding 1600 and 1400 evaluation trajectories, respectively, across the 10 tasks.

Evaluation protocol. For each task and displacement bin, we evaluate diagnostic-guided *steering*, with strength selected online by bisection, against a frozen-VLA *baseline* under identical target positions and scene layouts. An episode succeeds if it satisfies the LIBERO task-completion predicate. Each task–bin cell reports the fraction of successful episodes among 20 trials. Task-level totals pool all displacement bins, while aggregate results pool the ten tasks.

Implementation details. All experiments use OpenVLA and $\pi_{0.5}$ on NVIDIA RTX 4090D GPUs (PyTorch 2.2.0, Python 3.10, CUDA 12.1; MuJoCo 3.3.2 for simulation); no policy weights are modified in either condition, as steering is applied purely as an inference-time activation intervention on the frozen policy.

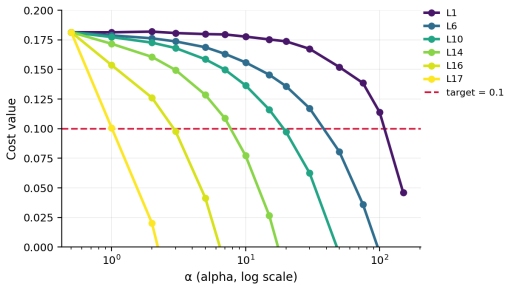


Figure 5: Last-layer predicted cost $\hat{c}_L(\alpha)$ vs. intervention strength α (log scale) at 7 representative layers of $\pi_{0.5}$, on in-distribution observations; dashed line marks a representative target c_{target}=0.10.

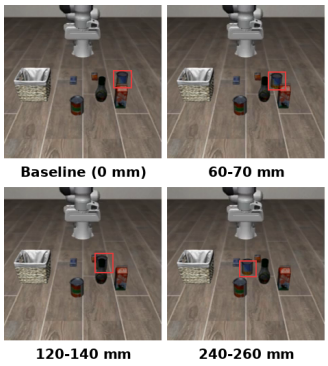


Figure 6: Visualization of object position perturbation in LIBERO simulation (*alphabet_soup* task, global camera view, frame at t=0); the red box marks the target object. The target soup can is displaced from its nominal position (top right) to increasingly large target displacements.

4.2 Main Out-of-Distribution Results

OpenVLA. Table 1 shows that steering raises aggregate success from 16.6% to 29.5% over 1,600 trials. At 0–20 mm, success improves from 76.0% to 92.0%; at intermediate offsets (20–60 mm), baseline falls from 39.5% to 10.0% while steering retains 64.5% to 33.0%. Beyond 60 mm, absolute margins narrow as both conditions struggle.

$\pi_{0.5}$. Table 2 shows aggregate success rising from 34.5% to 41.1% over 1,400 trials. Gains peak in the middle bins (+10 points at 160–180 mm, +9 at 180–200 mm); at 260–280 mm, both conditions approach 10%. Task-level gains range from a few points (salad_dressing, bbq_sauce) to over ten (cream_cheese, butter).

Both backbones show improvements in task-level totals across all ten tasks, despite substantial variation in baseline difficulty. The gains therefore extend across different target objects, while their magnitude remains task- and displacement-dependent. The narrowing margins at large displacements indicate that steering improves spatial robustness without eliminating the difficulty of severe distribution shifts.

4.3 Trigger-Frequency Sensitivity

We vary cooldown κ, EMA threshold τ, and slope threshold δ on $\pi_{0.5}$ for *cream_cheese* at 160–180 mm (layer 4; Figure 7). The intervention layer and strength-related settings remain fixed.

Cooldown separates successive perturbations, while cost and slope gates target high predicted cost that is not decreasing sufficiently. Steering therefore responds to the current rollout. Success peaks at moderate intervention frequency. The τ and δ sweeps show the full non-monotonic pattern; the tested κ range covers its rising half. These trends favor selective correction. This is a single-cell diagnostic.

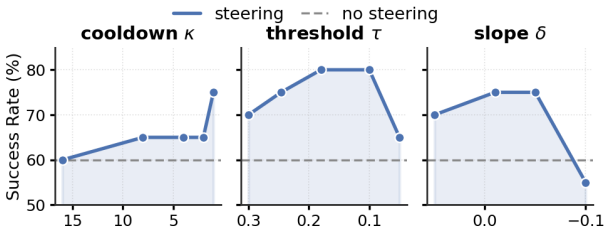


Figure 7: Success rate (%) vs. each trigger-frequency hyperparameter on *cream_cheese*, 160–180 mm ($\pi_{0.5}$, layer 4; n=20 per setting). Solid: steering; dashed: no-steering baseline (60%).

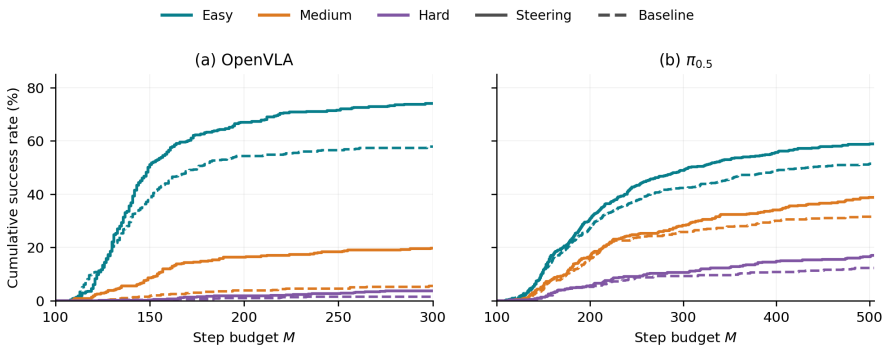


Figure 8: Cumulative success rate (%) vs. step budget M, pooled across all 10 LIBERO-Object tasks: (a) OpenVLA, easy/medium/hard $=$ 0–40/40–70/70–95 mm (n=400/400/800); (b) $\pi_{0.5}$, 140–200/200–240/240–280 mm (n=600/400/400); n denotes episodes per condition in each group. Solid: steering; dashed: baseline.

4.4 Success-Step Analysis

Figure 8 shows 404/1,600 OpenVLA steering successes by M=300 (25.3%), versus 265/1,600 for baseline (16.6%). The gain is largest in the easy and medium groups, while large displacements remain difficult. On $\pi_{0.5}$, the curves separate later across all three groups, indicating that steering continues to accumulate successful episodes as execution proceeds.

Table 3: Successful-episode completion steps across tasks and bins (steering / baseline).

Backbone	Successful episodes	Mean steps	Median steps
OpenVLA	472 / 265	214.6 / 151.9	150 / 141
$\pi_{0.5}$	576 / 483	233.3 / 223.3	204 / 199

The cumulative curves use all evaluated episodes as the denominator, whereas the completion-step statistics in Table 3 summarize successful episodes only. These complementary views describe how success accumulates over execution and when successful trajectories finish. Among successful episodes, steering has higher mean completion steps on both backbones, whereas the medians are closer. The two conditions succeed on different episode sets, so these conditional summaries are not an overall speed comparison. Together with the growing gaps in Figure 8, they are consistent with additional corrective rollouts reaching completion later in execution.

5 Conclusion

We propose an observe-and-steer framework that connects the interpretability of VLA hidden states with inference-time control. Our probing experiments show that end-effector affordance cost is linearly decodable across two distinct VLA backbones. Building on this finding, probe-direction steering uses the decoded signal to guide hidden-state interventions, improving closed-loop task success under object displacement without updating policy weights. The layer-wise analysis further shows that reducing predicted cost and changing actions are distinct requirements: effective steering must address both. Together, these results highlight affordance representations as a useful link between observing a policy's internal state and influencing its behavior.

Our evaluation focuses on simulated object-position shifts, and large displacements remain challenging. Extending steering beyond the grasping stage, evaluating broader distribution shifts, and validating the framework on physical robots are important directions for future work.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper and for retrieval and discovery of related work.

REFERENCES

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2023.

Hugo Buurmeijer, Carmen Amo Alonso, Aiden Swann, and Marco Pavone. Observing and controlling features in vision-language-action models. *arXiv preprint arXiv:2603.05487*, 2026.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Irving Fang, Juexiao Zhang, Shengbang Tong, and Chen Feng. From intention to execution: Probing the generalization boundaries of vision-language-action models. *arXiv preprint arXiv:2506.09930*, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *arXiv preprint arXiv:2506.09937*, 2025.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Conference on Robot Learning*, pp. 2743–2762. PMLR, 2025.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conference on Robot Learning*, pp. 540–562. PMLR, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.

Quanyi Li. Vlas are confined yet capable of generalizing to novel instructions. *arXiv preprint arXiv:2505.03500*, 2025.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, et al. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning*, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Yiran Pang, Yiheng Zhao, Zhuopu Zhou, Tingkai Hu, and Ranxin Hou. Is openvla truly robust? a systematic evaluation of positional robustness. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1–6, 2025.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. *arXiv preprint arXiv:2312.06681*, 2024.

Seongbin Park, Fan Zhang, Hossein Khalili, and Nader Sehatbakhsh. Adaptive safety probing for resource-efficient vision-language-action models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Siyu Xu, Zijian Wang, Yunke Wang, Chenghao Xia, Tao Huang, and Chang Xu. Affordance field intervention: Enabling vlas to escape memory traps in robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 37206–37215, 2026.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics. *arXiv preprint arXiv:2406.10721*, 2024.

Fan Zhang, Seongbin Park, Baharan Mirzasoleiman, Shahriar Talebi, and Nader Sehatbakhsh. Probeact: Probe-guided training-free failure recovery in vision-language-action models. *arXiv preprint arXiv:2606.09740*, 2026a.

Jiachen Zhang, Junnan Nie, Junyi Lao, Wei Cheng, Chenghao Liu, Jiaxin Jiang, and Songfang Huang. What frozen vlas already know about success: A probing study of value-like structure in foundation robot policies. *arXiv preprint arXiv:2605.28527*, 2026b.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Supplementary Material

A.1 Implementation Details

Table 4 lists the final gating and strength-search hyperparameters used for both backbones in the main experiments (Tables 1–2 of the main text), using the notation defined there (κ, τ, δ, W, α_{ema} for gating; α/bisection for strength selection; c_{target} and tolerance for the strength-search stopping criterion). Both backbones denote the intervention magnitude by α, following the main text; the two backbones' α values are not on a shared numeric scale. Cooldown κ and slope window W are counted in simulation steps for OpenVLA (which intervenes at every step) and in replans for $\pi_{0.5}$ (one replan every 5 simulation steps).

Table 4: Final hyperparameters used in the main OOD experiments.

Parameter	OpenVLA	$\pi_{0.5}$
Cooldown κ	5 (steps)	1 (replan)
Threshold τ	0.264	0.246
EMA rate α_{ema}	0.3	0.3
Slope δ	-0.01	-0.01
Slope window W	10 (steps)	10 (replans)
Strength search range	$[0, 200]$	$[0, 200]$
Bisection iterations (max)	6	6
Cost target c_{target}	0.1	0.1
Tolerance band	± 0.01	± 0.02
Readout layer L	31	17
Replan interval	every step	every 5 steps

Cost-target calibration. Let p_q denote the q-th percentile. τ is calibrated offline from the empirical distribution of terminal-layer probe predictions on *successful, unperturbed* baseline rollouts, restricted to the first 40% of each episode (a proxy for the pre-grasp approach stage): $\tau{=}p_{75}$ of this pooled early-stage distribution, with tolerance $=\frac{1}{2}(p_{50}{-}p_{25})$. For $\pi_{0.5}$, this gives $p_{25}{\approx}0.20$, $p_{50}{\approx}0.22$, $p_{75}{\approx}0.246$, so we set $\tau{=}0.246$. For OpenVLA, applying the same procedure to 500 in-distribution baseline rollouts (446 successful) gives $p_{25}{\approx}0.19$, $p_{50}{\approx}0.23$, $p_{75}{\approx}0.264$, so we set $\tau{=}0.264$.

c_{target} is set from a different statistic of the same successful rollouts: the terminal-layer cost *at the final step*, once the task is essentially complete, rather than the early-stage window used for τ. This near-completion cost is markedly lower than the early-stage distribution above and concentrates in a similar band for both backbones ($p_{25}{\approx}0.11$, median$\approx$0.12 for $\pi_{0.5}$; $p_{25}{\approx}0.11$, median$\approx$0.13 for OpenVLA). We set $c_{\text{target}}{=}0.10$ for both backbones, at the lower edge of this shared near-completion band, so the strength-search stopping criterion targets the cost level a successful rollout actually reaches once nearly finished, rather than the roughly twice-as-large level typical of its early approach stage; by this same logic the early-stage p_{25} itself (e.g. 0.20 for $\pi_{0.5}$) is a poor target, and indeed produced steering pushes too weak to recover failing rollouts when tried in preliminary runs. Because this near-completion level is shared by both backbones' successful rollouts, $c_{\text{target}}{=}0.1$ transfers unchanged across backbones despite their different τ.

A.2 Layer Selection Diagnostic

A layer is useful for steering only if perturbing it *simultaneously* (a) moves the terminal-layer probe prediction $\hat{c}_L$ toward c_{target} and (b) induces a non-trivial change in the emitted action: layers that satisfy only (a) can drive the probe's own readout arbitrarily low without the policy moving differently, and layers that satisfy only (b) change the action without a reliable cost signal to gate on. Intervention sensitivity varies by up to two orders of magnitude with depth, so we sweep every layer across its full range of tested magnitudes α rather than comparing layers at one shared value, and identify which (layer, α) combinations satisfy both conditions at once. This narrows the per-task layer search in the main experiments to a compact, backbone-specific neighborhood rather than scanning the full depth for every task. Because $\pi_{0.5}$ emits continuous flow-matched actions while OpenVLA emits discretized action tokens, the two backbones require different action-change

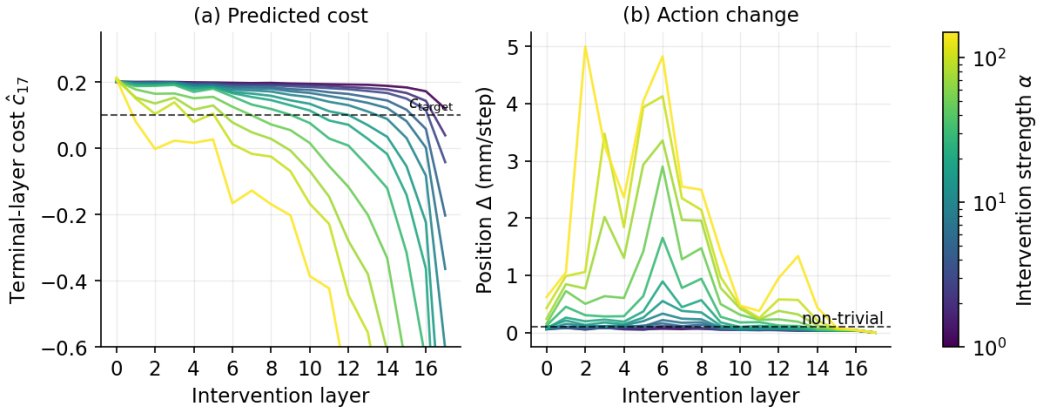


Figure 9: $\pi_{0.5}$: terminal-layer cost (a) and end-effector position change (b) vs. intervention layer, one line per tested α (log-scale colorbar). Dashed lines mark c_{target} and the non-trivial action threshold.

measurements (physical end-effector displacement for $\pi_{0.5}$; a normalized-unit action difference for OpenVLA, whose 256-bin discretization means sub-bin changes read as exactly zero) and are reported separately below.

$\pi_{0.5}$

Figure 9 sweeps all 18 decoder layers across the full tested magnitude grid ($\alpha \in \{0, 1, 2, 3, 5, 7, 10, 15, 20, 30, 50, 75, 100, 150\}$), plotting the terminal-layer (L=17) predicted cost (a) and the resulting end-effector position change (b) against layer depth, one line per tested α. L0 is excluded from the analysis below: it is the only layer where increasing α moves predicted cost in the wrong direction, so it never approaches c_{target}. Table 5 lists, for every other layer, which grid points bring $\hat{c}_{17}$ within 0.05 of c_{target}=0.1 *and* the position change to at least 0.1 mm/step, and how many such points exist. Qualifying points concentrate in L1–L12, with L9 and L8 satisfying the most (4 and 3 of 14 tested magnitudes, respectively); no layer beyond L12 satisfies both conditions at any tested magnitude.

Table 5: $\pi_{0.5}$: layers with at least one grid point jointly satisfying $\hat{c}_{17} \approx c_{\text{target}}$ and non-trivial position change, and the α values at which this holds.

Layer	Qualifying α	Count
L9	15, 20, 30, 50	4
L8	20, 30, 50	3
L1	100, 150	2
L2	75, 100	2
L4	75, 100	2
L5	75, 100	2
L6	50, 75	2
L7	30, 50	2
L3	100	1
L10	30	1
L11	20	1
L12	20	1

OpenVLA

Figure 10 sweeps all 32 decoder layers across a 10-point magnitude grid ($\alpha \in \{5, 10, 20, 35, 50, 75, 100, 150, 200, 300\}$) on a fixed in-distribution observation (*alphabet_soup*, initial timestep), plotting the terminal-layer (L=31) predicted cost (a) and the translation-channel action change (b) against layer depth, in the same style as Figure 9. Table 6 lists, using the same criteria

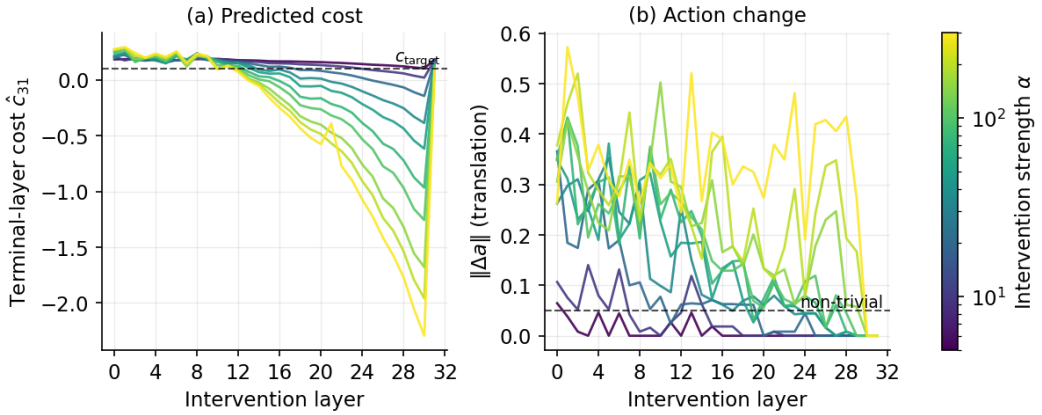


Figure 10: OpenVLA: terminal-layer cost (a) and translation-channel action change (b) vs. intervention layer, one line per tested α (log-scale colorbar). Dashed lines mark c_{target} and the non-trivial action threshold.

as for $\pi_{0.5}$ ($\hat{c}_{31}$ within 0.05 of c_{target} and $\|\Delta a\| \geq 0.05$), which layers have at least one qualifying α. Qualifying layers span L5, L7, L10–L19, with L11 the most robust (satisfying 7 of the 10 tested magnitudes) followed by L12 (5 of 10); no layer outside this range satisfies both conditions at any tested magnitude.

Table 6: OpenVLA: layers with at least one tested α jointly satisfying $\hat{c}_{31} \approx c_{\text{target}}$ and non-trivial action change, and how many of the 10 tested magnitudes qualify.

Layer	Qualifying α	Count
L11	35, 50, 75, 100, 150, 200, 300	7
L12	75, 100, 150, 200, 300	5
L7	75, 100, 150, 200	4
L10	100, 150, 200, 300	4
L13	35, 50, 75	3
L14	20, 35, 50	3
L15	20, 35, 50	3
L16	20, 35, 50	3
L5	35	1
L17	20	1
L18	20	1
L19	20	1

Discussion

Both backbones show the same pattern: cost and action are jointly controllable across the shallower-to-mid portion of the network (L1–L12 of 18 for $\pi_{0.5}$; roughly L5–L19 of 32 for OpenVLA), while the deepest layers can drive the probe's self-reported cost arbitrarily low without a corresponding change in behavior. We use this joint diagnostic to prioritize candidate layers for the closed-loop OOD evaluation in Tables 1–2 of the main text.

A.3 Task Suite Generation

Both backbones' displacement bins (Tables 1–2 of the main text) are generated by the same procedure, applied within each backbone's own displacement range.

Deterministic replay. Each LIBERO scene is fully described by a 110-dimensional MuJoCo state snapshot (joint angles/velocities, target and receptacle position and orientation, and the po-

sition of every other object), read via `env.get_sim_state()`. Displacing the target to a sampled offset only edits the two coordinates of this snapshot corresponding to the target's position; `env.set_init_state()` then restores the edited snapshot exactly, with no other source of randomness. This makes every trial bit-exact reproducible and, in particular, guarantees that the baseline and steering conditions evaluated on the same task and bin (Evaluation protocol, main text) see *identical* target positions and scene layouts—the displacement suite is generated once and both conditions replay the same stored snapshots.

Reachability-constrained sampling regions. The ten LIBERO-Object tasks are split into two groups according to how far the end effector's reachable workspace allows the target to move in each direction. Five tasks (*alphabet_soup, ketchup, butter, milk, chocolate_pudding*) are displaced within a quarter-disk, offset (dx, dy) with $dx \geq 0$ and $dy \geq 0$ relative to the nominal position (i.e., only away from the arm and toward one lateral side); the other five (*cream_cheese, salad_dressing, bbq_sauce, tomato_sauce, orange_juice*) sit closer to the arm's reach limit in the away direction and are instead displaced within a half-disk, $dx \leq 0$ with $dy \in [-r, r]$ (toward the arm, free to move laterally on either side). Displacing either group beyond its allotted angular range risks placing the target outside the arm's reachable workspace.

Stratified radial sampling. Within each displacement bin $[r_{\min}, r_{\max}]$, positions are drawn by stratified sampling: the radial interval is split into n_{rings} rings and the (quarter- or half-) angular range into n_{angles} sectors, and one position is drawn uniformly at random from each of the $n_{\text{rings}} \times n_{\text{angles}}$ ring–sector cells. Both backbones use n_{rings}=2, n_{angles}=10, giving 20 positions (and thus 20 trials) per task per bin. OpenVLA's eight reported bins cover 0–95 mm; $\pi_{0.5}$'s seven reported bins cover 140–280 mm, a larger range consistent with it being displaced farther before failing (main text).

Handling placement conflicts at large displacements. At the larger displacements used for $\pi_{0.5}$, the newly displaced target can end up too close to one of the fixed non-target objects for LIBERO's placement sampler to find a non-overlapping layout. Following the convention used by LIBERO-PRO's own extended-displacement suites, each sampled position is first attempted with the original object layout; only if that specific position triggers a placement failure do we identify the nearest conflicting non-target object and move it out of frame, retrying until the layout succeeds. The target's own position is never altered by this process, and in practice only a small minority of sampled positions require moving any object at all (the rest keep the full original layout). This does not arise for OpenVLA, whose reported displacement range stays within the region where the original layout is always feasible.